Variants of the AES Key Schedule for Better Truncated Differential Bounds

Patrick Derbez[1⋆], Pierre-Alain Fouque[1⋆⋆], Jérémy Jean[2], Baptiste Lambin[1⋆⋆⋆]

[1] Univ Rennes, CNRS, IRISA
baptiste.lambin,patrick.derbez@irisa.fr
pierre-alain.fouque@univ-rennes1.fr
[2] ANSSI, France
Jeremy.Jean@ssi.gouv.fr

Abstract. Differential attacks are one of the main ways to attack block ciphers. Hence, we need to evaluate the security of a given block cipher against these attacks. One way to do so is to determine the minimal number of active S-boxes, and use this number along with the maximal differential probability of the S-box to determine the minimal probability of any differential characteristic. Thus, if one wants to build a new block cipher, one should try to maximize the minimal number of active S-boxes. On the other hand, the related-key security model is now quite important, hence, we also need to study the security of block ciphers in this model.

In this work, we search how one could design a key schedule to maximize the number of active S-boxes in the related-key model. However, we also want this key schedule to be efficient, and therefore choose to only consider permutations. Our target is AES, and along with a few generic results about the best reachable bounds, we found a permutation to replace the original key schedule that reaches a minimal number of active S-boxes of 20 over 6 rounds, while no differential characteristic with a probability larger than 2^{-128} exists. We also describe an algorithm which helped us to show that there is no permutation that can reach 18 or more active S-boxes in 5 rounds. Finally, we give several pairs (P_s, P_k), replacing respectively the `ShiftRows` operation and the key schedule of the AES, reaching a minimum of 21 active S-boxes over 6 rounds, while again, there is no differential characteristic with a probability larger than 2^{-128}.

Keywords: AES · Key schedule · Related-key · Truncated Differential

⋆ Patrick Derbez was supported by the French Agence Nationale de la Recherche through the CryptAudit project under Contract ANR-17-CE39-0003.

⋆⋆ Pierre-Alain was supported by the French Agence Nationale de la Recherche through the BRUTUS project under Contract ANR-14-CE28-0015.

⋆⋆⋆ Baptiste Lambin was supported by the Direction Générale de l'Armement (Pôle de Recherche CYBER).

This paper was accepted at SAC 2018
https://doi.org/10.1007/978-3-030-10970-7_2

1 Introduction

First introduced in 1991 by Biham and Shamir [2], differential cryptanalysis is one of the main tool to analyze and attack symmetric primitives. The main idea is to introduce some differences in the plaintext, and see how these differences propagate through the different steps of the algorithm, independently from the key. For example, given an encryption function $\mathcal{E}(p, k)$ encrypting the plaintext $p \in \mathbb{F}_2^{n_b}$ using a key $k \in \mathbb{F}_2^{n_k}$, if one is able to prove that there exists a pair of differences $\Delta_{in}, \Delta_{out} \in \mathbb{F}_2^{n_b}$ such that $\mathcal{E}(p \oplus \Delta_{in}, k) = \mathcal{E}(p, k) \oplus \Delta_{out}$ for all keys, then it gives a strong distinguisher for the encryption function $\mathcal{E}$. Moreover, due to the non-linearity of $\mathcal{E}$, such a differential relation could only hold with a certain probability. Consequently, a lot of work has been put into designing algorithms that search for the best possible differential characteristics of a given cipher. For instance, Matsui's algorithms [18] were the first designed. Most of modern ciphers are now built as *iterated ciphers*, i.e., a round function f is built and repeated several times, XOR-ing a round key between each application of f, see Figure 1. Thus, to search for such a pair $(\Delta_{in}, \Delta_{out})$, one often studies the propagation of the input difference through each round of the cipher, thus leading to a *differential characteristic* consisting of all differences in each state s_i.

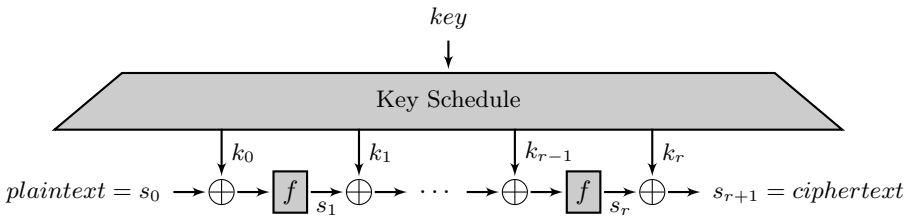

Fig. 1: Generic iterated cipher construction

One can also choose to consider only *truncated differences*, that is, only look at whether or not the difference in one byte is zero. While this can also directly lead to various attacks, e.g., impossible differential attacks [1, 16], it can also be used to get some results in differential cryptanalysis. Indeed, in most cipher designs, the non-linear component consists of an S-box, a small non-linear function applied several times over all iterations. This S-box is the reason that some differential characteristic only holds with a certain probability. Given an S-box S acting on a small number of s bits, and for each pair $(\Delta_{in}, \Delta_{out}) \in \mathbb{F}_2^{2s}$, one can easily compute how many $x \in \mathbb{F}_2^s$ verifies the relation $S(x \oplus \Delta_{in}) = S(x) \oplus \Delta_{out}$. This allow to compute the Difference Distribution Table (DDT) of the S-box, which gives the probability that the above relation holds for each $(\Delta_{in}, \Delta_{out})$. Thus, given a differential characteristic, one can easily compute the probability that it holds, simply by multiplying all differential probabilities of each S-box

together.[3] Hence, given a *truncated* differential characteristic, while we cannot determine the exact probability that this characteristic holds, we can deduce its minimal probability. Indeed, if the S-box has a maximal differential probability of p, and there are n S-boxes with a non-zero difference (called *active S-boxes*), then the truncated differential characteristic holds with a probability at most p^n. Thus, given the maximal differential probability of the S-box used and the bit-length n_k of the key, one can easily deduce the minimal number of active S-boxes n_{min} that leads to $p^{n_{min}} < 2^{-n_k}$. So, if for a given number of round, we can prove that there is at least n_{min} active S-boxes, we know that there would be no differential characteristic with a probability better than 2^{-n_k}, which would mean that finding a pair of plaintexts satisfying this characteristic would *a priori* costs more than an exhaustive search for the key.

Such differentials and truncated differentials can also be considered in the *related-key model*. First introduced in 2009 to attack AES-192 and AES-256 [3, 4], this model allows the attacker to inject differences in the plaintext, but also in the key. Another worth-mentionning model is the more recent *related-tweak model* for tweakable block ciphers, where the attacker fully controls an additional input for the block cipher called a *tweak* [17, 21]. While this model is closer to chosen-plaintext attacks, the tweak is often (but not necessarily) used alongside the key and thus involved in the key schedule, such as in the TWEAKEY framework [13]. Since the attacker can now inject some differences in both the plaintext and the key, this causes a large increase in the complexity to search differential and truncated differential characteristics. Nonetheless, several tools have been designed to tackle this problem [5, 9, 10]. Hence, a few proposals were made to give another, more secure, key schedule for some primitives, such as [19, 7] for AES and [20] for `SKINNY` and AES-based constructions from FSE 2016 [12]. However, their main concern was mostly to design a more secure key schedule, without considering the possible loss in efficiency. To that regard, Khoo et al. [14] proposed a new key schedule for AES which consists in only a permutation at the byte level, based on their proof on the number of active S-boxes in the related-key model for AES. Using a permutation thus leads to a very efficient key schedule, both in software and hardware, and can also make the analysis easier.

Our Contributions. In this paper, we go further and study how we can design a *good* permutation to use as the key schedule in AES-128. More precisely, we first start by giving some bounds on the reachable minimal number of active S-boxes for up to 7 rounds of AES if we use a simple permutation as its key schedule. Especially, we show that there is no permutation that can reach a minimal number of active S-boxes of 18 or more over 5 rounds. These bounds allow us to know the results that a "perfect" permutation could reach. Then, we provide a method to search for such a permutation. To do so, we reused the meta-heuristic approach given by Nikolić in [20], combined with a Constraint

[3] Using the fair assumption that each round is independent, which while obviously not true, is admitted as a reasonable assumption.

Programming model inspired from the work of Gerault et al. in [10]. Especially, we give a way to model the underlying equations of a truncated differential characteristic, leading to a more precise model than the original one from [10]. Namely, the truncated differential characteristics found are always valid unless we consider the DDT of the S-box.

We also went further and modified both the key schedule and one step of the AES round function (namely, `ShiftRows`) to see whether we can achieve better bounds. As a result, we exhibit a permutation P_k which, when used as the AES key schedule, lead to a minimal number of active S-boxes of 20 over 6 rounds, while no characteristic has a probability larger than 2^{-128}. When changing both the key schedule and the `ShiftRows` step, we give several pairs of permutations (P_k^i, P_s^i) that have a minimal number of active S-boxes of 21 over 6 rounds, while again, no characteristic has a probability larger than 2^{-128}. While we applied this method to AES, it is quite generic and could also be used on any block cipher, as long as one have an efficient enough way to compute the minimal number of active S-boxes. Our implementation is available at `https://github.com/TweakAESKS/TweakAESKS`.

2 Background

Differential cryptanalysis was first introduced by Biham and Shamir in 1991 [2] and mainly consists in studying the propagation of differences between two plaintexts through the cipher. Here, we only consider *truncated differences*, that is, we are only interested in whether a byte does have a non-zero difference (*active* byte) or not (*inactive* byte). Our work is centered around AES, for which we make a few remainders. AES is the NIST block cipher standard, derived from Rinjdael [8]. It uses an internal state of 128 bits, and several key sizes are available, namely 128, 192 and 256. Here, when mentioning AES, we refer to the 128-bit version.

It is an SPN block cipher, iterating a round function $R = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SB} \circ \mathsf{ARK}$ 10 times, where each component of the round function is quickly described in the following. The state can be viewed as a 4×4 byte array, and thus we will often talk about *columns* of the state. The round function consists in four operations: `AddRoundKey` (ARK), `SubBytes` (SB), `ShiftRows` (SR) and `MixColumns` (MC). ARK XORs the round key into the internal state. This round key is derived from the master key using a key schedule KS, for which we do not give details, our ultimate goal being to change it. We refer the interested reader to [8] for the original descriptions. SB applies a non-linear operation (called S-box) on each byte of the state, then SR performs a cyclic shift of each row, where Row j is shifted by $j-1$ bytes to the left, $j \in \{1, 2, 3, 4\}$. Finally, MC is a linear operation that multiplies each column of the internal state by an MDS matrix with coefficients in $\mathbb{F}_{2^8}$.

We first recall several well known properties of the MC operation, which will be used is the rest of the article. Here, $w(x)$ correspond to the number of active bytes in x, which is either a state or a column of the state.

Proposition 1 (`MixColumns` MDS property). *Let z and y be two state columns such that $\mathsf{MC}(z) = y$. Then, either $w(z)+w(y) = 0$ or $w(z)+w(y) \geq 5$. Moreover, for any five bytes in y and z, there exists one linear equation between those five bytes.*

Proof. This comes directly from the fact that the matrix used in the MC operation is MDS.

Proposition 2 (`MixColumns` linear property). *Let z, z', y, y' be four state columns such that $\mathsf{MC}(z) = y$ and $\mathsf{MC}(z') = y'$. Then, the `MixColumns` MDS property also holds for $(z \oplus z')$ and $(y \oplus y')$, that is: either $w(z \oplus z') + w(y \oplus y') = 0$ or $w(z \oplus z') + w(y \oplus y') >= 5$*

Proof. This comes directly from the previous proposition and the fact that MC is linear.

Lemma 1. *Let k, x, y, z be four state columns such that $\mathsf{MC}(z) = y$, z contains at least one active byte and $x = y \oplus k$. Denote by $i_{y,z}$ the number of inactive bytes in y and z (i.e., $i_{y,z} = 8 - w(y) - w(z)$) and $c_{z,k,x}$ the number of bytes from z that are cancelled by k in x. If $i_{y,z} + c_{y,z,k} \geq 5$, then there is at least one linear equation on some bytes of k. Moreover, this can only happens if $c_{y,z,k} \geq 2$.*

Proof. If $i_{y,z} + c_{y,z,k} \geq 5$, then from the `MixColumns` MDS property, it follows that there is an equation between any five bytes chosen from the inactive ones in y and z, and the bytes from z which are cancelled by k. If we denote such a cancelled byte by z_i, that is, $z_i \oplus k_i = 0$, then we have $k_i = z_i$, hence the equation involves some bytes of k and some inactive bytes from y and z, which are zeros.

Since z contains at least one active byte, we have $w(z) + w(y) \geq 5$, hence $i_{y,z} \leq 3$. Therefore, if $c_{y,z,k} = 1$ (i.e., only one byte if cancelled), we have $i_{y,z} + c_{y,z,k} \leq 4$, and thus no equation is implied.

When considering truncated differentials, we are often interested in the number of active S-boxes, that is, the number of active bytes going through an S-box (i.e., active bytes at the beginning of the round). We will often refer to the (minimal) number of active S-boxes in a characteristic as the *length* of the characteristic, and to a *minimal characteristic* to refer to a characteristic which reaches the minimal number of active S-boxes. Given a truncated differential characteristic of length n, one can deduce the maximal probability that this characteristic can have once being instantiated. Indeed, if the S-box has a maximal non-zero differential probability of p, then the maximal probability of this characteristic is p^n. If one studies a block cipher with a key of length n_k bits, then the goal is to prove that no characteristics can be instantiated with a probability larger than 2^{-n_k}. Hence, for AES, since the maximal differential probability of the S-box is 2^{-6}, we know that if for a given number of rounds the minimal number of active S-boxes is greater or equal than 22, then no differential characteristic with a differential probability larger than 2^{-128} exists.

Searching whether a characteristic reaching a given length or maximal probability exists has been a major focus in academic research. One way to find the best probability is to proceed in two steps. First, one try to find a truncated differential characteristic with a minimal number of active S-boxes, and then try to instantiate this characteristic. When searching for such a truncated differential characteristic, one can choose to consider additional information about the cipher along with "basic" propagation rules coming from the round function, to avoid trying to instantiate characteristics that would not be instantiable anyway. Hence for AES, we give the following definitions.

Definition 1. *A characteristic is said to be valid in the "truncated differential setting" if and only if the* `MixColumns` *linear property is always verified and there is at least one non-trivial solution to the system of equations (if any) induced by Lemma 1.*

A characteristic that remains valid even when one does not consider the `MixColumns` *linear property nor the equations is said to be valid in the* pure truncated differential setting.

The point of these definitions is twofold. On the one hand, since the pure truncated differential setting contains significantly less constraints, the minimal characteristic could be a lot easier to find. However, it may result in an invalid characteristic when one tries to instantiate it, which could have been detected in the truncated differential setting. Conversely, finding the minimal characteristic in the truncated differential setting could be harder, but the only thing that could invalidate this characteristic is the S-box DDT.

We chose to use the same approach as Gerault et al. [10], who proposed to use two Constraint Programming models. The first one was used to find the minimal characteristics for AES, considering only the `MixColumns` linear property. The second one takes a list of truncated characteristic and tries to find the best instantiation (if any) of each characteristic with respect to its probability. As we aim at changing the key schedule, we changed these models, detailed in the following.

Model 1 *This model takes as input a permutation P_k to use as the key schedule and a number of rounds, and output the minimal number of active S-boxes with these parameters in the truncated differential setting. Compared to the first model of [10], we directly model the equations coming from the* `MixColumns` *operation (see Lemma 1), resulting in a more reliable result, albeit being slower. We refer the reader to Appendix A for the method used to model these equations.*

Model 2 *This model also takes as input a permutation P_k for the key schedule and a number of rounds, along with a list of truncated differential characteristics. It then goes through each of these truncated characteristics, and tries to find a instantiation with a probability larger than 2^{-128}. If such an instantiation is found, it gives its probability and the differential characteristic, otherwise it just stops without trying to find an instantiation with a probability smaller than 2^{-128}.*

3 Generic Bounds

Before trying to find a permutation that reaches a certain number of active S-boxes, we need to study which number of S-boxes we can reach. From the fact that using a permutation as the key schedule implies that the number of active bytes in the key is constant, we can deduce several bounds on the number of active S-boxes. To demonstrate these bounds, we show that there is always a differential characteristic of a certain length, independently from the permutation used in the key schedule.

Proposition 3. *Using a permutation as the key schedule, there is always a differential characteristic of length 1 (resp. 5). for 2 (resp. 3) rounds. For 4 rounds, there is always a characteristic of length either 8, 9 or 10. Moreover, these differential characteristics always remain valid in the truncated differential setting.*

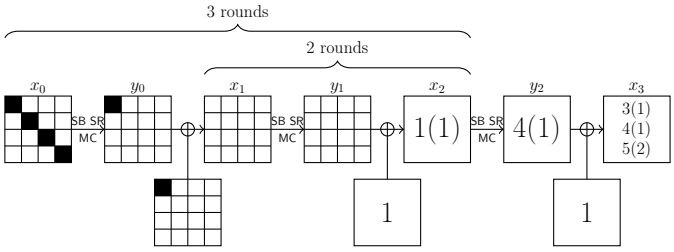

Fig. 2: Characteristic always valid for 2,3 and 4 rounds. $x(y)$ means that there are x active Sboxes somewhere in the state, with y columns containing at least one active bytes. Multiple $x(y)$ in a state means that one of them must be true

Proof. Such a characteristic is depicted in Figure 2. For 2 rounds, there is only one active byte in the second state, which is cancelled by the active byte in the key. For 3 rounds, the previous characteristic is extended by adding one more round before it, and comes directly from the `MixColumns` MDS property.

For 4 rounds, we add one more round after the 3-round differential characteristic. Since y_2 has four active bytes on the same column, and since the key has one active byte anywhere in the key state, x_3 can have either 3, 4 or 5 active bytes, which results in a differential characteristic of length either 8, 9 or 10.

No equation is implied since there is always at most one active key byte that is cancelled with the ARK operation for each round (Lemma 1). Finally, there are only two `MixColumn` transitions with active bytes, one of the form $\mathsf{MC}(z) = y$ where z and y are one column of the state with $w(z) = 4$ and $w(y) = 1$ and another of the form $\mathsf{MC}(z') = y'$, where $w(z') = 1$ and $w(y') = 4$. Hence, $w(z \oplus z') \geq 3$ and $w(y \oplus y') \geq 3$, and thus the `MixColumns` linear property is always valid.

Corollary 1. *Using a permutation as the key schedule, the optimal bounds on the number of active S-boxes that can be proven for 2, 3 and 4 rounds is respectively 1, 5 and 10 in the truncated differential setting.*

The proof of this corollary comes directly from the previous proposition. If we try to extend the previous characteristic with one more round, we obtain that there is always a characteristic of length either 19, 20, 21, 24 or 25 in the truncated differential setting. However, if we only consider the pure truncated differential setting, then we have the following proposition.

Proposition 4. *For 5, 6 and 7 rounds, there is always a characteristic of length respectively 14, 18 and 21 in the pure truncated differential setting.*

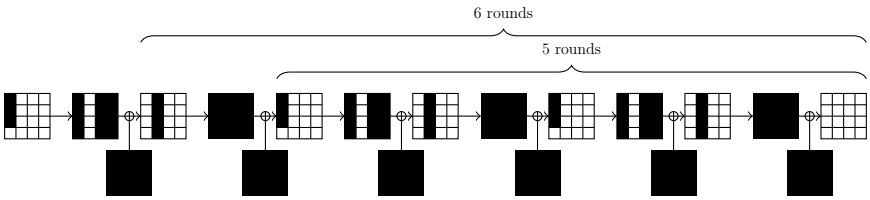

Fig. 3: Characteristic always valid for 5, 6 and 7 rounds.

Proof. Such a characteristic is depicted in Figure 3. Note that considering how this kind of characteristic is built, there are a lot of underlying equations in the truncated differential setting, which is very likely to make this characteristic invalid. However, in the pure differential setting, these characteristics always remains valid as they come directly from the propagation rules of the AES round function.

Corollary 2. *Using a permutation as the key schedule, the optimal bounds on the number of active S-boxes that can be proven for 5, 6 and 7 rounds is respectively 14, 18 and 21 in the pure truncated differential setting.*

Now the first question that we may ask is whether or not there exists a permutation which reaches all those bounds. Fortunately, such a permutation was already found by Khoo et al. in [14], which is

$$P_{KLPS} = \big(5 \;\; 2 \;\; 3 \;\; 8 \;\; 9 \;\; 6 \;\; 7 \;\; 12 \;\; 13 \;\; 10 \;\; 11 \;\; 0 \;\; 1 \;\; 14 \;\; 15 \;\; 4\big)\,.$$

However, if we study this permutation in the truncated differential setting for 7 rounds using Model 1, then we have that the minimum number of active S-boxes becomes 22, proving that no differential characteristic with a probability larger than 2^{-128} can be found, hence the following theorem.

Theorem 1. *We can find a permutation for the key schedule which guarantees that no differential characteristic with a probability larger than 2^{-128} exists for 7 or more rounds of AES. Moreover, this does not depend on the S-box DDT.*

Obviously, now the main question is: How far can we go? Can we find a permutation that reach 22 S-boxes for 6 rounds or lower, or at least a permutation such that no differential characteristic with probability larger than 2^{-128} exists? This would allow us to show that even with an extremely simple and efficient key schedule, we can still have a rather good security against differential attacks in the related-key model. We study this in the next section.

4 Searching for a Permutation

4.1 Bound on 5 Rounds

In this section, we show that there is no permutation that can reach a minimal number of active S-boxes of 18 over 5 rounds. While this does not imply that we cannot find a permutation such that there is no differential characteristic with a probability better than 2^{-128}, this still gives us a good idea of what we can reach for 5 rounds.

To achieve this, we proceed in two steps. First, we search for a set of cycles such that using a given cycle of this set, one cannot build a truncated differential characteristic of length strictly lower than 18, which induces equations (according to Lemma 1) on at most 1 round. Since all permutations can be decomposed into a composition of cycles, this would not only speed up the search (since we do not need to check every permutation one at a time), but also gives a way to build all permutations that *could* reach 18 S-boxes on 5 rounds. To build such a set of cycles, we used a quite straightforward algorithm.

First, we suppose that the cycle starts with 0. Then, we guess the image of 0, and for each of those guesses, we have two cases: either the cycle is not complete, and thus we need to make another guess on the next element of the cycle, or the cycle is closed. Whenever we make a new guess or decide that the cycle is closed, we can build several truncated key characteristics $k_0 \to k_1 \to ... \to k_4$ according to the current (partial) cycle examined: each active byte in this truncated key characteristic must be a byte that belongs to the current (partial) cycle. Then, for each of those truncated key characteristics, we search the minimal number of active S-boxes that we can reach using this characteristic. To speed up the search, we only consider truncated characteristics that induces equations on at most 1 rounds, such that these characteristics are always valid in the truncated differential model. If, for a given (partial) cycle, one can find a corresponding truncated characteristic with strictly less than 18 S-boxes, then we know that this (partial) cycle cannot be part of the permutation we are looking for. If we were in the case where the cycle was not complete, then we know that we do not need any more guesses, and if the cycle was closed, we can dismiss it. Thus in the end, we will have a set of closed cycles which start with 0, and for which all truncated characteristics that induces equations on at most 1 rounds have

at least 18 active S-boxes. We then need to apply the same algorithm, but this time with cycles beginning by 1 and not containing 0 (to avoid repetitions) and so on.

In the end, we have a set of permutations for which we know that, if a permutation reaches a minimal number of active S-boxes of 18 (or higher), then it must be built from this set of cycles. Thus, we just need to built all possible permutations from these cycles, and plug them into Model 1 to see if the actual minimal number of S-boxes is indeed 18 or higher. The number of cycles which can be used to build a permutation reaching 18 S-boxes is given in Appendix B, and by testing all possible combinations, we found out that there is no such permutation, hence the following theorem.

Theorem 2. *There is no permutation that, when used as key schedule, can reach a minimal number of active S-boxes of 18 or higher over 5 rounds. Using the same method, we were also able to find at least one permutation which have a minimal number of active S-boxes of 16 over 5 rounds, namely:*

$$\left(15\ \ 0\ \ 2\ \ 3\ \ 4\ \ 11\ \ 5\ \ 7\ \ 6\ \ 12\ \ 8\ \ 10\ \ 9\ \ 1\ \ 13\ \ 14\right).$$

However, the possibility of reaching 17 S-boxes over 5 rounds is still unknown, and the complexity of the algorithm for 6 rounds is too high. Hence, we focused our search for a permutation reaching 22 active S-boxes over 6 rounds, using another approach we detail in the next section.

4.2 Finding a Permutation over 6 Rounds

First of all, let us take a quick look at how we could naively search for such a permutation. This is rather straightforward: for each possible permutation, we check whether the minimal number of S-boxes is at least 22. Since we are looking for a permutation over 16 bytes, we have $16! > 2^{44}$ possible permutations. While 2^{44} basic operations could be achievable in a reasonable amount of time, the computation of the minimal number of S-boxes is actually quite costly. For example, if one would use the algorithm from [9] which has an approximate complexity of 2^{34} operations, this would raise the total cost to 2^{78} operations, which is clearly impractical. While we do not have a complexity estimation for our constraint programming tool, the average time to solve Model 1 is about 40 minutes for 6 rounds, which would lead to way too much time to try each permutation, so exhausting all permutations is clearly not a viable way to proceed.

On the other hand, one could try to pick a random permutation, evaluate its minimal number of S-boxes, and try again if this number is lower than 22. While the cost of computing the minimal number of S-boxes remains, this approach could be successful if the density of the set of permutation reaching 22 S-boxes overall permutations is high enough. Indeed, if we do this for 7 rounds, we are able to find a permutation reaching the same number of S-boxes for 7 rounds and lower as the permutation from [14] in about 200 tries. However, this approach was not able to find a permutation reaching 22 S-boxes over 6 rounds.

Hence, we need something more efficient for 6 rounds. Inspired by the work of Nikolić [20], we choose to use a meta-heuristic called *simulated annealing*. Meta-heuristics are a class of search algorithms which aim to find an (almost) optimal solution to an optimization problem, often inspired by some real-life phenomenon. To be more precise, unlike *Constraint Programming* or *Integer Linear Programming* which aims at recovering an *optimal* solution, meta-heuristics only look for a *good enough* solution: it may not be optimal, but it should be rather close to an optimal solution. In our case, we could define our optimization problem as: Which permutation maximize the minimal number of active S-boxes over 6 rounds? However, we are not really interested in maximizing the minimal number of S-boxes, we only need to find a permutation which reaches 22 S-boxes. Moreover, our problem is of the form *"Maximize the minimum value of a given function"*, which is not something easily handled by classical techniques like Constraint or Linear Programming. Finally, meta-heuristics are designed to be both relatively easy to implement and rather efficient, hence they seem quite appropriate to tackle this problem.

We give a generic algorithm for simulated annealing in Appendix C, also given in [20]. The main idea of this algorithm is to try to maximize a function $f(x)$ (called *objective function*) by progressively improving a solution, starting from a random one, while allowing degradation. To be more precise, starting from a random x_0, the algorithm builds another solution x_i from x_{i-1} using the function ϵ. Then, if $f(x_i) > f(x_{i-1})$, then x_i is accepted and the algorithm continues. However, if $f(x_i) \leq f(x_{i-1})$, which would mean that x_i is worse than the previous solution x_{i-1}, x_i is only accepted with some probability depending on a value T, and if it is rejected, another x_i is generated from x_{i-1}. Then, the value T is updated with a function $\alpha(T)$. For more details about this algorithm and the choice of its parameters, we refer the reader to [20, 6, 15].

Now, we need to see how we implement this algorithm in practice. As in [20], we did not observe major differences between different parameters for the initial temperature T_0 and the cooling schedule $\alpha(T)$. Hence, we only give one set of parameters, from which all our following results come from. For the initial temperature, we used $T_0 = 2$. For the cooling schedule, we used the same one as in [20], i.e., $\alpha(T) = \frac{T}{1+\beta T}$ with $\beta = 0.001$ Finally, the neighbor function ϵ generates a new permutation from the one that has been tested. This new permutation should be "close" to the previous one, hence we use a random transposition to generate a new permutation, namely, $\epsilon(x) = \tau \circ x$ where τ is a random transposition.

The only thing missing to implement the algorithm is a way to evaluate $f(x)$. Recall that in our case, $f(x)$ is the minimal number of active S-boxes for a given permutation x. A naive way to compute $f(x)$ would be to solve Model 1 with the permutation x. However, as mentioned before, solving this model is quite costly, which would results in a very slow meta-heuristic. Instead, we make the following observation. Let n be the number of active S-boxes we want to prove, that is, we want to find a permutation for which the minimal number of active S-boxes is at least n. Then, given a certain permutation, we are only interested in one

fact: does this permutation have a characteristic with a length strictly less than n? If so, then even if this characteristic is not a minimal one, we still know that this permutation will not reach our goal of a minimum of n active S-boxes. This allows to slightly modify the original algorithm for a much quicker execution, which lead to more permutation being evaluated and thus better chances to find a good one. The complete algorithm is given as Algorithm 1, with a more detailed explanation below.

Algorithm 1 Tweaked Simulated Annealing

Input: Target length n
$x \leftarrow$ *random permutation*, $T \leftarrow 2$, $l \leftarrow 0$
while $l < n$ **do**
  $\tau \leftarrow$ *random transposition*, $x' \leftarrow \tau \circ x$
  $l' \leftarrow \texttt{quicksearch}(x', n)$
  if $l' \geq n$ **then**
    $x \leftarrow x'$, $l \leftarrow \texttt{fullsearch}(x)$
  else if $l' > l$ **then**
    $x \leftarrow x'$, $l \leftarrow l'$
  else
    $r \leftarrow U[0,1]$ *Generate a uniformly random real number in [0,1]*
    if $r < e^{\frac{l'-l}{T}}$ **then**
      $x \leftarrow x'$, $l \leftarrow l'$
    end if
  end if
  $T \leftarrow \frac{T}{1+0.001T}$
end while
Output: x

So instead of directly computing the minimal number of active S-boxes for a given permutation, we do the following. We first use the algorithm `quicksearch`, which is a classical dynamic programming algorithm which, given a permutation x and a target number of S-boxes n, search for a *relatively* short characteristic of length $\leq n$. As mentioned before, the idea is to use the fact that we are mostly interested in whether or not a characteristic of length strictly less than n exists. This algorithm performs this relatively quickly, without having to find *the* minimal number of S-boxes. Once we get such a characteristic of length l', three cases can happen.

- If $l' \geq n$, then the permutation might be a good one. However, since the `quicksearch` algorithm does not return the length of the shortest characteristic, we need to call the `fullsearch` algorithm, which basically solves Model 1 using the provided permutation, and returns the real minimal num-

ber of S-boxes. If the output of `fullsearch` is greater or equal than n, then we found a permutation and the algorithm terminates. If not, we still choose to update x to x', because the fact that `quicksearch` returned a value greater or equal than n means that the permutation looked quite good at first glance. We also update l to the real minimal number of active S-boxes of x, since otherwise the algorithm would terminate while it did not found a permutation reaching n S-boxes.
- Otherwise if $l' > l$, that is, the permutation x' seems to have a minimal number of S-boxes greater than the previous one, then we update x to x' too. This corresponds to the case $f(x') > f(x)$ in the original Simulated Annealing algorithm.
- Finally, if $l' \leq l$, this is the same as the original algorithm. We accept the solution x' and update x to it only with a certain probability depending on the current temperature T and the respective number of S-boxes found for x and x'.

We first launched this algorithm using $n = 20$, and were able to find the permutation P_k (given below) reaching this minimal number of S-boxes in about 2^{16} tries:

$$P_k = \big(8\ \ 1\ \ 7\ \ 15\ \ 10\ \ 4\ \ 2\ \ 3\ \ 6\ \ 9\ \ 11\ \ 0\ \ 5\ \ 12\ \ 14\ \ 13\big)\,.$$

Reaching 21 S-boxes is still an open question and for reference, we were able to test about 2^{24} permutations in several days. However, we were able to show that using P_k as the key schedule, while only reaching a minimum amount of 20 S-boxes in the truncated setting, still guarantee that no characteristic with a probability better then 2^{-128} can be found when one use the DDT of the AES S-box. To do that, we used Model 2, which allows to check if there is a characteristic with a better probability than 2^{-128} and to exhibit one if that is the case. To make this model work, we need to give it a list of truncated differential characteristics, and it will check if such a characteristic can be instantiated with a probability better than 2^{-128}. Hence, to prove that P_k has no such characteristic, we need a list of all valid truncated characteristics of 20 and 21 S-boxes (since 22 S-boxes already guarantees that no characteristic will be instantiable with a probability better than 2^{-128}). This can be computed rather quickly using Model 1 and asking the solver to find all characteristics of length 20 and 21. There are 253 characteristics of length 20 and 3284 of length 21. After about nine hours on a standard desktop to loop through all these characteristics, it turns out that none of them can be instantiated[4] with a probability better than 2^{-128}. In conclusion, we were able to find a permutation P_k such that using this permutation as the key schedule of AES-128 guarantees that no differential characteristic with a probability better than 2^{-128} exists over 6 or more rounds. For reference, we also ran Model 1 on this permutation to get the minimal number of active S-boxes for a lower amount of rounds, summarized in Table 1.

[4] For reference, the best probability we could reach among all the characteristics of length 20 was 2^{-134}

Number of rounds	2	3	4	5	6	7
Original key schedule	1	3	9	11	$13^{\dagger}$	15
P_{KLPS}	1	5	10	14	$18^{\dagger}$	22
P_k	1	5	10	15	$20^{\dagger}$	23

Table 1: Minimal number of S-boxes that our permutation P_k reaches on a given number of rounds compared to the one from [14]. †No instantiation with a better probability than 2^{-128}.

Now, even if we were able to find a permutation leading to no differential characteristic of probability better than 2^{-128} for 6 rounds or more, it still only reaches 20 S-boxes in the truncated setting. Hence, we would like to see if by modifying further the AES round function, we could reach more active S-boxes. This is treated in the next section.

5 Tweaking Both `ShiftRows` and the Key Schedule

Using the approach given in the previous section allowed to find a permutation for the key schedule, which induces a minimal number of S-boxes of 20 for 6 rounds. Here, we would like to see if by changing the `ShiftRows` operation in the AES-128, we could reach a better number of active S-boxes, namely 21 or 22. Obviously, we cannot try all possible permutations for `ShiftRows`, as again, there are 2^{44} permutations over 16 elements. Hence, we show here how we restricted ourselves to only a few thousand candidates for `ShiftRows`, which are the most likely to lead to a good minimal number of active S-boxes, and give a few examples of pairs (P_s, P_k) that reach 21 S-boxes for 6 rounds, where P_s is used instead of the `ShiftRows` operation, and P_k instead of the original key schedule KS of AES.

First, we can see that we can drastically reduce the number of candidates for P_s using the following two propositions. We denote $\mathcal{P}_i$ the set of all permutations P_i acting insides the columns of the state, i.e., there exists four permutations $P_i^0, P_i^1, P_i^2, P_i^3$ over four elements such that P_i^j acts on the j-th column and $P_i = P_i^0 \circ P_i^1 \circ P_i^2 \circ P_i^3$, and $\mathcal{P}_c$ the set of all permutations which permutes the columns of the state.

Proposition 5. *Let P_s and P'_s be two permutations over 16 elements such that $P'_s = P'_i \circ P_s \circ P_i$, where $P_i, P'_i \in \mathcal{P}_i$, and let $P'_k = P_i^{-1} \circ P_k \circ P_i$. Then using (P'_s, P'_k) instead of (SR, KS) will lead to the same minimal number of active S-boxes that using (P_s, P_k) instead of (SR, KS). Hence, we can build equivalence classes $\mathcal{E}_i(P_s) = \{P'_s \mid \exists\, P_i, P'_i \text{ s.t. } P'_s = P_i \circ P_s \circ P'_i\}$, and there are 10147 such equivalence classes.*

Proof. We need to show that, for each characteristic we can build using (P_s, P_k), one can find a characteristic with the same number of active S-boxes using (P'_s, P'_k), where $P'_s = P'_i \circ P_s \circ P_i$ and $P'_k = P_i^{-1} \circ P_k \circ P_i$.

Given a characteristic $(X_0, \ldots, X_r)$ such that the length of the characteristic is given by $\sum_{i=0}^{r} X_i$, and denote Y_i the state after the MC operation such that $X_{i+1} = Y_i \oplus K_i$. We have $Y_{i+1} = \mathsf{MC} \circ P_s \circ \mathsf{SB}(Y_i \oplus K_i)$ and $K_{i+1} = P_k(K_i)$, where P_k is a bytewise permutation. For all i, let $K'_i = P_i^{-1}(K_i)$ and $Y'_i = P_i^{-1}(Y_i)$, hence we have

$$\begin{aligned} K'_{i+1} &= P_i^{-1}(K_{i+1}) = P_i^{-1} \circ P_k(K_i) \\ &= P_i^{-1} \circ P_k \circ P_i \circ P_i^{-1}(K_i) \\ &= P'_k \circ P_i^{'-1}(K_i) \\ &= P'_k(K'_i). \end{aligned}$$

So P'_k is a valid key schedule. Furthermore, note that when considering the propagation of active bytes through MC, one only need to consider the number of active bytes before MC in one given columns to know the number of active byte after MC in that same column. Hence, since $P_i \in \mathcal{P}_i$ only permutes bytes inside each column, the number of active bytes does not change in each column and thus for any $P_i \in \mathcal{P}_i$, MC and $\mathsf{MC}' = \mathsf{MC} \circ P_i$ behave similarly when searching for truncated differential characteristics, i.e., replacing MC by MC' has no effect. In the same way, one can replace MC by $P_i \circ \mathsf{MC}$ with $P_i \in \mathcal{P}_i$. Moreover, SB acts on each byte separately, hence $P_i \circ \mathsf{SB} = \mathsf{SB} \circ P_i$. Thus, we have:

$$\begin{aligned} Y'_{i+1} &= P_i^{-1}(Y_i) = P_i^{-1} \circ \mathsf{MC} \circ P_s \circ \mathsf{SB}(Y_i \oplus K_i) \\ &= P_i^{-1} \circ \mathsf{MC} \circ P_s \circ \mathsf{SB}(P_i \circ P_i^{-1}(Y_i) \oplus P_i \circ P_i^{-1}(K_i)) \\ &= \mathsf{MC} \circ P_s \circ \mathsf{SB}(P_i(Y'_i) \oplus P_i(K'_i)) \quad \textit{replacing } P_i^{-1} \circ \mathsf{MC} \textit{ by } \mathsf{MC} \textit{ has no effect} \\ &= \mathsf{MC} \circ P'_i \circ P_s \circ P_i \circ SB(Y'_i \oplus K'_i) \quad \textit{replacing } \mathsf{MC} \textit{ by } \mathsf{MC} \circ P'_i \textit{ has no effect} \\ &= \mathsf{MC} \circ P'_s \circ \mathsf{SB}(Y'_i \oplus K'_i). \end{aligned}$$

So (P'_s, P'_k) correctly defines a round function and we have $X'_{i+1} = Y'_i \oplus K'_i = P_i^{-1}(Y_i \oplus K_i) = P_i^{-1}(X_{i+1})$ for all i. Hence, each X'_i is a permutation of X_i, and thus the corresponding characteristic $(X'_0, \ldots, X'_r)$ has the same number of active S-boxes as $(X_0, \ldots, X_r)$.

Proposition 6. *Let P_s and P'_s be two permutations over 16 elements such that $P'_s = P_c^{-1} \circ P_s \circ P_c$ where $P_c \in \mathcal{P}_c$, and let $P'_k = P_c^{-1} \circ P_k \circ P_c$. Then, using (P'_s, P'_k) instead of (SR, KS) will lead to the same minimal number of active S-boxes that using (P_s, P_k) instead of (SR,KS). Hence we can combine this with the previous proposition, and for each class representative P_s of some class $\mathcal{E}_i(P_s)$ defined previously, we can build equivalence classes $\mathcal{E}(P_s) = \{P'_s \mid \exists\, P_c \in \mathcal{P}_c \text{ s.t. } P'_s = P_c^{-1} \circ P_s \circ P_c\}$, and there are 9186 such equivalence classes.*

The proof of the previous theorem is very similar to the proof of Proposition 5 and is given in Appendix D. Hence, we only need to consider 9186 possible candidates P_s to replace SR, instead of 2^{44}. Moreover, we would like to avoid weakening AES in the single-key model. In that model, the original `ShiftRows`

allows to reach full diffusion after 3 rounds. So we only considered the permutations that also reached full diffusion in at most 3 rounds, and there are 4381 of them. Finally, recall that in the pure truncated differential setting, using the original `ShiftRows` implies that there is always a characteristic of length 18 which is built using a fully active key. While this characteristic has high chances of being invalidated once we consider the equations it implies on the key, we still would like to avoid it. To do that, we used the following proposition.

Proposition 7. *If one uses a permutation P_s instead of `ShiftRows` such that P_s send the bytes from any one column to at most three columns, then the characteristic from Proposition 4 cannot happen.*

Proof. The characteristic from Proposition 4 can be built because a state containing a single fully active column lead to a fully active state after $\mathsf{MC} \circ \mathsf{SR}$. However, if one uses a permutation P_s which send the bytes from any one column to at most three columns, then the state after $\mathsf{MC} \circ P_s$ will contain at most 3 fully active column. Thus, when XOR-ing the key afterwards, the resulting state would have at least 4 active bytes, instead of 3 in the characteristic from Proposition 4, thus this characteristic cannot happen.

Hence, we only want to try some permutations P_s instead of `ShiftRows` which verify the previous propositions and achieve a full diffusion in at most 3 rounds in the single-key model, which lead to 3288 possible candidates for P_s. Now everything is quite straightforward. We reuse Algorithm 1 to search for a permutation leading to 21 S-boxes, except that we use a different permutation than `ShiftRows` in the `quicksearch` algorithm and modified Model 1 to use that permutation instead of `ShiftRows` for the `fullsearch` algorithm. We also added the additional condition that it should stop after 24 hours if no permutation reaching the objective was found. Surprisingly, the `quicksearch` algorithm ran faster with those permutations than with the original SR, which allowed us to test about 2^{25} permutations P_k on average in 24 hours for a specific candidate P_s. After a few more than 100 possible P_s tried, we were able to find several pairs (P_s, P_k) that reach 21 S-boxes (see Appendix E). After testing about 1100 candidates for P_s, finding a pair (P_s, P_k) that reaches 22 S-boxes is still an open problem.

We also used Model 2, tweaked to use a different permutation instead of SR, to check if there is a differential characteristic with a probability better than 2^{-128} over 6 rounds with these pairs (P_s, P_k), and again, none of these permutation allows such a characteristic.

6 Conclusion

In this paper, we studied how AES would behave in the related-key model if we change its key schedule to a much simpler and efficient one, namely a permutation. We first gave a few generic bounds about the best number of active S-boxes reachable for a given number of round, and especially, we showed that

no permutation can reach a minimal number of 18 or more active S-boxes over 5 rounds. However we were able to exhibit a permutation reaching 16 S-boxes over 5 rounds, hence closing the gap a bit further. We showed that we can find a permutation which allows to have at least 20 active S-boxes over 6 rounds, while guaranteeing that no characteristic with a probability larger than 2^{-128} exists. This allows us to reach the same amount round than with the original AES-128 key schedule (see [9]), but with a more efficient key schedule which is also easier to analyze and has a higher minimal number of active S-boxes. We also took a look at how modifying the SR operation could improve the minimal number of S-boxes over 6 rounds. It turns that we can find several pairs (P_s, P_k) to use instead of SR and the key schedule (respectively) which allows to have at least 21 S-boxes over 6 rounds, and again, no characteristic with a probability better than 2^{-128}. We also provided a Constraint Programming model which can handle directly the equations coming from `MixColumns`, thus allowing to find the exact minimal number of active S-boxes considering everything but the S-box DDT in a reasonable amount of time and memory. Our implementation is available at `https://github.com/TweakAESKS/TweakAESKS`.

A few open questions remain. First, could we reach a minimal number of 22 active S-boxes changing only the key schedule (and possibly SR) for 6 rounds? In the same idea, could we close the gap for 5 rounds? We know that we cannot get 18 or more active S-boxes, but 16 S-boxes is reachable, thus the possibility of reaching 17 S-boxes is still unknown. Finally, we chose to change the SR operation, but how about changing either MC or the S-box? While changing everything would lead to a cipher that does not have much in common with AES, it could answer the following generic question: Can we build an AES-like SPN (with a round function structured as $\mathsf{MC} \circ P_s \circ \mathsf{SB}$ where P_s is a permutation and MC uses an MDS matrix) using a permutation as the key schedule, which could reach either 22 S-boxes over 6 rounds, or guarantee that no characteristic with probability better than 2^{-128} exists over 5 rounds?

References

1. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: International Conference on the Theory and Applications of Cryptographic Techniques, Springer (1999) 12–23
2. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of CRYPTOLOGY **4**(1) (1991) 3–72
3. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Matsui, M., ed.: Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Volume 5912 of Lecture Notes in Computer Science., Springer (2009) 1–18
4. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and Related-Key Attack on the Full AES-256. In Halevi, S., ed.: Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Volume 5677 of Lecture Notes in Computer Science., Springer (2009) 231–249

5. Biryukov, A., Nikolic, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In Gilbert, H., ed.: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. Volume 6110 of Lecture Notes in Computer Science., Springer (2010) 322–344
6. Černỳ, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of optimization theory and applications **45**(1) (1985) 41–51
7. Choy, J., Zhang, A., Khoo, K., Henricksen, M., Poschmann, A.: AES Variants Secure against Related-Key Differential and Boomerang Attacks. In Ardagna, C.A., Zhou, J., eds.: Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings. Volume 6633 of Lecture Notes in Computer Science., Springer (2011) 191–207
8. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1999)
9. Fouque, P., Jean, J., Peyrin, T.: Structural Evaluation of AES and Chosen-Key Distinguisher of 9-Round AES-128. In Canetti, R., Garay, J.A., eds.: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Volume 8042 of Lecture Notes in Computer Science., Springer (2013) 183–203
10. Gérault, D., Lafourcade, P., Minier, M., Solnon, C.: Revisiting AES Related-Key Differential Attacks with Constraint Programming. IACR Cryptology ePrint Archive **2017** (2017) 139
11. Jean, J.: TikZ for Cryptographers. `https://www.iacr.org/authors/tikz/` (2016)
12. Jean, J., Nikolic, I.: Efficient Design Strategies Based on the AES Round Function. In Peyrin, T., ed.: Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Volume 9783 of Lecture Notes in Computer Science., Springer (2016) 334–353
13. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. (2014) 274–288
14. Khoo, K., Lee, E., Peyrin, T., Sim, S.M.: Human-readable Proof of the Related-Key Security of AES-128. IACR Trans. Symmetric Cryptol. **2017**(2) (2017) 59–83
15. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598) (1983) 671–680
16. Knudsen, L.: DEAL-a 128-bit block cipher. (1998)
17. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings. IACR Transactions on Symmetric Cryptology **2017**(3) (2017) 37–72
18. Matsui, M.: On Correlation Between the Order of S-boxes and the Strength of DES. In Santis, A.D., ed.: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. Volume 950 of Lecture Notes in Computer Science., Springer (1994) 366–375
19. Nikolic, I.: Tweaking AES. In Biryukov, A., Gong, G., Stinson, D.R., eds.: Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers. Volume 6544 of Lecture Notes in Computer Science., Springer (2010) 198–210

20. Nikolic, I.: How to Use Metaheuristics for Design of Symmetric-Key Primitives. In Takagi, T., Peyrin, T., eds.: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III. Volume 10626 of Lecture Notes in Computer Science., Springer (2017) 369–391
21. Zong, R., Dong, X., Wang, X.: MILP-Aided Related-Tweak/Key Impossible Differential Attack and Its applications to QARMA, Joltik-BC. Cryptology ePrint Archive, Report 2018/142 (2018) https://eprint.iacr.org/2018/142.

A Modelizing the MC equations in Constraint Programming

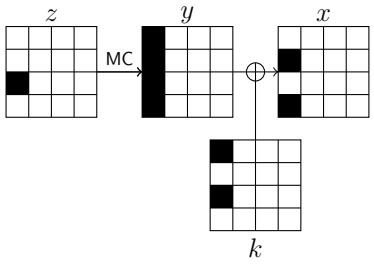

Fig. 4: A partial round that implies one equation

We will give here an example as how we generate constraints to modelize the equations coming from the MC operation. From the MDS property of MC, we know that there is en equation between any set of five bytes taken from the same column of z and y. Specifically, we have the following equation, where coefficient are in $\mathbb{F}_{256}$:

$$5.z[0] + 7.z[1] + z[3] = 2.y[0] + y[2].$$

Now we take the situation given in Fig. 4. First, all bytes 0,1 and 3 of z are inactive, hence we can replace $z[0], z[1]$ and $z[3]$ in the previous equation by zeros. Moreover, we can see that both $y[0]$ and $y[2]$ are cancelled by some bytes in k, i.e. $y[i] \oplus k[i] = 0, i \in \{0, 2\}$. Hence, our equation becomes $2.k[0] + k[2] = 0$.

So, if this situation occurs, we know that we have a specific equation involving bytes of k. However, this equation has coefficient in $\mathbb{F}_{256}$, which are not handled by Constraint Programming solvers. Hence, we modelize this equation at a bit-level, using the fact that the scalar multiplication in $\mathbb{F}_{256}$ corresponds to a linear

operation in $\mathbb{F}_2^8$. By denoting $k_j^i, j \in [0,7], i \in 0,2$ the j-th bit of $k[i]$, we have

$$\begin{pmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} k_0^0 \\ k_1^0 \\ k_2^0 \\ k_3^0 \\ k_4^0 \\ k_5^0 \\ k_6^0 \\ k_7^0 \end{pmatrix} + \begin{pmatrix} k_0^1 \\ k_1^1 \\ k_2^1 \\ k_3^1 \\ k_4^1 \\ k_5^1 \\ k_6^1 \\ k_7^1 \end{pmatrix} = 0.$$

We now have everything to modelize this case using an *if-constraint*. In our model, we have a binary variable for each byte of the state which is set to 0 if the corresponding byte is inactive, and 1 otherwise. Since all the equations only involves some key-bits, we also have binary variables for each bit of each subkey. Restricting this in the situation given in Fig. 4, we would have binary variables $z[i], y[i], x[i], k[i], i \in [0,15]$ modelizing whether or not bytes are active, and binary variables $k_j^i, i \in [0,15], j \in [0,7]$ for each bit of the key. Obviously, we need to modelize the fact that if a key byte is inactive, then its bits are all zeros, which is easily modelized with

$$k[i] = 0 \iff k_j^i = 0 \; \forall j \in [0,7].$$

Hence, the above equation only holds when $z[0] = z[1] = z[2] = 0, y[0] = y[2] = 1$ and $x[0] = x[2] = 0$. Note that we do not need to check that $k[0] = 1$ since the fact that $y[0] = 1$ and $x[0] = 0$ necessarily implies that $k[0] = 1$ (and the same argument goes for $k[2]$). So, to modelize this case, we use an *if-constraint*. Such a constraint is of the form $E \Rightarrow C$, and means that if the expression E is true, then the constraint C must hold. Thus, we modelize the above situation with the constraint

$$\begin{array}{c} z[0] = 0 \wedge z[1] = 0 \wedge z[2] = 0 \wedge y[0] = 1 \\ \wedge\, y[1] = 1 \wedge x[0] = 0 \wedge x[2] = 0 \end{array} \Rightarrow \begin{array}{rl} k_7^0 + k_0^1 & = 0 \bmod 2 \wedge \\ k_0^0 + k_7^0 + k_1^1 & = 0 \bmod 2 \wedge \\ k_1^0 + k_2^1 & = 0 \bmod 2 \wedge \\ k_2^0 + k_7^0 + k_3^1 & = 0 \bmod 2 \wedge \\ k_3^0 + k_7^0 + k_4^1 & = 0 \bmod 2 \wedge \\ k_4^0 + k_5^1 & = 0 \bmod 2 \wedge \\ k_5^0 + k_6^1 & = 0 \bmod 2 \wedge \\ k_6^0 + k_7^1 & = 0 \bmod 2 \end{array}$$

Hence in our model, we need to do this for all rounds and for each column of the state. The number of constraints coming from this is easy to compute. For a fixed round and column, denote i the number of inactive bytes taken in

z, hence $\binom{4}{i}$ possibilities, with $1 \leq i \leq 3$. Denote j the number of inactive bytes taken in y hence $\binom{4}{j}$ possibilities. Hence, we have $5-i-j$ active bytes (that are cancelled) in y, taken in the remaining $4-j$ bytes, thus $\binom{4-j}{5-i-j}$ possibilites. Moreover, we know from Lemma 1 that we must have $5-i-j \geq 2$. So the number of constraints for a fixed round and a fixed column is

$$\sum_{i=1}^{3} \sum_{j=0}^{3-i} \binom{4}{i} \binom{4}{j} \binom{4-j}{5-i-j} = 164,$$

hence $656r$ constraints for r rounds.

B Number of cycles to build a permutation reaching 18 S-boxes over 5 rounds

If one would want to build a permutation reaching 18 active S-boxes over 5 rounds, then Table 2 gives the number of possible cycle which can be used to build such a permutation. For example, this table means that if the permutation contains a cycle of length 11, then there are only 48 cycles of this length which can be used to build the permutation. This table also implies that the permutation should not contain a cycle of length ≥ 12. As mentioned in Section 4.1, none of the possible combinations of those cycles allows to build a permutation reaching 18 active S-boxes over 5 rounds.

Length of the cycle	Number of cycles
1	16
2	120
3	796
4	6576
5	25656
6	78448
7	112608
8	74904
9	15576
10	1344
11	48

Table 2: Number of cycles which must be use to build a permutation reaching 18 A-boxes over 5 rounds

C Generic simulated annealing algorithm

Algorithm 2 Simulated Annealing [20]

Input: initial temperature T_0, cooling schedule $\alpha(T)$, neighbor function $\epsilon(x)$

$x \leftarrow random, \quad T \leftarrow T_0$
while termination criteria not met **do**
  $x' \leftarrow \epsilon(x)$
  if $f(x') > f(x)$ **then**
    $x \leftarrow x'$
  else
    $r \leftarrow U[0,1]$ *Generate a uniformly random real number in [0,1]*
    if $r < e^{\frac{f(x')-f(x)}{T}}$ **then**
      $x \leftarrow x'$
    end if
  end if
  $T \leftarrow \alpha(T)$
end while

Output: x

D Proof of Proposition 6

As in the proof of Proposition 5, we need to show that, for each characteristic we can build using (P_s, P_k), one can find a characteristic with the same number of active S-boxes using (P'_s, P'_k), with $P'_s = P_c^{-1} \circ P_s \circ P_c$ and $P'_k = P_c^{-1} \circ P_k \circ P_c$, $P_c \in \mathcal{P}_c$.

Given a characteristic $(X_0, \ldots, X_r)$, and using the same notation as in the the proof of Proposition 5, for all i let $K'_i = P_c^{-1}(K_i)$ and $Y'_i = P_c^{-1}(Y_i)$. Showing that P'_k is a valid key-schedule is done in the same way as for Proposition 5. Furthermore, note that since MC acts on each column separately, we have $\mathsf{MC} \circ P_c^{-1} = P_c^{-1} \circ \mathsf{MC}$. In the same way, SB acts on each byte separately, hence $P_c \circ \mathsf{SB} = \mathsf{SB} \circ P_c$. Thus we have

$$\begin{aligned} Y'_{i+1} &= P_c^{-1}(X_{i+1}) = P_c^{-1} \circ \mathsf{MC} \circ P_s \circ \mathsf{SB}(Y_i \oplus K_i) \\ &= P_c^{-1} \circ \mathsf{MC} \circ P_s \circ \mathsf{SB}(P_c(Y'_i) \oplus P_c(K'_i)) \\ &= P_c^{-1} \circ \mathsf{MC} \circ P_s \circ \mathsf{SB} \circ P_c(Y'_i \oplus K'_i) \\ &= \mathsf{MC} \circ P_c^{-1} \circ P_s \circ P_c \circ \mathsf{SB}(Y'_i \oplus K'_i) \\ &= \mathsf{MC} \circ P'_s \circ \mathsf{SB}(Y'_i \oplus K'_i) \end{aligned}$$

So again, (P'_s, P'_k) correctly defines a round function and $X'_{i+1} = P_c^{-1}(X_{i+1})$ for all i. Thus each X'_i is a permutation of X_i, hence the corresponding characteristic $(X'_0, \ldots, X'_r)$ has the same number of active S-boxes as the characteristic $(X_0, \ldots, X_r)$.

E Pairs (P_s, P_k) reaching 21 Sboxes over 6 rounds

(P_s, P_k)	# **iterations**
$P_s^1 = (0\ 1\ 2\ 4\ 3\ 8\ 9\ 12\ 5\ 13\ 14\ 15\ 6\ 7\ 10\ 11)$ $P_k^1 = (10\ 4\ 12\ 11\ 6\ 2\ 5\ 1\ 8\ 0\ 9\ 7\ 13\ 14\ 15\ 3)$	$3151253 \sim 2^{21.6}$
$P_s^2 = (0\ 1\ 2\ 4\ 3\ 8\ 9\ 12\ 5\ 6\ 13\ 14\ 7\ 10\ 11\ 15)$ $P_k^2 = (15\ 14\ 11\ 10\ 6\ 12\ 4\ 0\ 3\ 8\ 1\ 9\ 2\ 5\ 13\ 7)$	$42414349 \sim 2^{25.3}$
$P_s^3 = (0\ 1\ 4\ 8\ 9\ 10\ 12\ 13\ 5\ 6\ 14\ 15\ 2\ 3\ 7\ 11)$ $P_k^3 = (14\ 12\ 8\ 6\ 7\ 4\ 0\ 1\ 3\ 11\ 10\ 2\ 9\ 5\ 13\ 15)$	$8588115 \sim 2^{23}$
$P_s^4 = (0\ 1\ 2\ 8\ 4\ 9\ 12\ 13\ 5\ 6\ 7\ 14\ 3\ 10\ 11\ 15)$ $P_k^4 = (12\ 14\ 11\ 4\ 8\ 0\ 3\ 7\ 10\ 15\ 2\ 9\ 6\ 13\ 5\ 1)$	$15016901 \sim 2^{23.8}$
$P_s^5 = (0\ 1\ 2\ 8\ 4\ 9\ 12\ 13\ 3\ 5\ 14\ 15\ 6\ 7\ 10\ 11)$ $P_k^5 = (5\ 9\ 15\ 13\ 3\ 4\ 6\ 2\ 11\ 7\ 10\ 0\ 8\ 14\ 1\ 12)$	$51700477 \sim 2^{25.6}$

Table 3: Pairs (P_s, P_k) which reach 21 S-boxes, along with the number of P_k tried before founding it

For a given P_s^i, we also took a look at the permutations P'_k that are rather "close" to the ones we found, that is, permutations P'_k which are one or two transpositions away from each P_k^i. It turns out that, except for (P_s^4, P_k^4), none of these permutations also reach 21 S-boxes. Oddly, there are 3 permutations that are 1 transposition away from P_k^4 which also reach 21 S-boxes when using P_s^4 instead of SR , and again, none of them has a differential characteristic with a probability better than 2^{-128} over 6 rounds. Those three permutations are

$$P_k^{4'} = \big(14\ \ 12\ \ 11\ \ 4\ \ 8\ \ 0\ \ 3\ \ 7\ \ 10\ \ 15\ \ 2\ \ 9\ \ 6\ \ 13\ \ 5\ \ 1\big),$$
$$P_k^{4''} = \big(12\ \ 14\ \ 11\ \ 4\ \ 10\ \ 0\ \ 3\ \ 7\ \ 8\ \ 15\ \ 2\ \ 9\ \ 6\ \ 13\ \ 5\ \ 1\big),$$
$$P_k^{4'''} = \big(12\ \ 14\ \ 11\ \ 4\ \ 8\ \ 0\ \ 3\ \ 7\ \ 2\ \ 15\ \ 10\ \ 9\ \ 6\ \ 13\ \ 5\ \ 1\big).$$